LISTING OF CLAIMS

1. (Currently Amended) A symmetric Finite Impulse Response (FIR) filter providing linear scalability ~~and implementation without the need for delay lines~~, comprising:

a multiprocessor architecture wherein each processor includes an ALU ~~including a plurality of ALUs~~ (Arithmetic and Logic Unit), a Multiplier unit ~~Multipliers units~~, Data cache, and Load/Store unit, the multiprocessor architecture ~~units~~ sharing a common Instruction cache ~~[[,]]~~ and a multi-port memory, and

~~an~~ assigning means for assigning to each processor ~~available processing unit~~ the computation of specified unique FIR partial product terms and the accumulation of ~~each~~ computed partial product terms without the need for delay lines in order to generate filtered ~~on specified~~ output sample values.

2. (Canceled).

3. (Currently Amended) A method for ~~implementing an improved~~ symmetric Finite Impulse Response (FIR) filtering of input data ~~filter~~ providing linear scalability using a multiprocessor ~~multiprocessing~~ architecture platform ~~without the need for delay lines~~, comprising ~~the steps of~~:

assigning to each processor ~~available processing unit~~ the computation of specified unique FIR partial product terms and the accumulation of ~~; and accumulating each~~ computed partial product terms without use of a delay line to generate filtered ~~on specified~~ output ~~sample~~ values.

4. (Currently Amended) The method of claim 3 wherein the multiprocessor ~~multiprocessing~~ architecture comprises a two processor ~~2 parallel processor~~ architecture, the method comprising:

initializing a loop-index by zero;

loading an input data term ~~sample~~ for a first processor;

loading a filter coefficient for the first processor;

multiplying the input data term ~~sample~~ and filter coefficient in the first processor;

updating a current output in the first processor;

updating a partial output, if required, in the first processor;

loading an input data term ~~sample~~ for a second processor;

loading a filter coefficient for the second processor;

multiplying the input data term ~~sample~~ and filter coefficient in the second processor;

updating a current output in the second processor;

updating a partial output, if required, in the second processor;

increment the loop-index by 2; and

stopping if the loop-index equals end, else, returning to loading the input data term ~~sample~~ for the first processor and repeating.

5. (Currently Amended) The method of claim 3, comprising:

Initializing a loop index by zero;

loading an input data term ~~sample~~ and filter coefficients;

multiplying the input data term ~~sample~~ and filter coefficient;

updating a current output;

updating a partial output, if required;

incrementing the loop index ~~by a number of processors~~; and

if loop termination is condition satisfied then:

stopping; else

returning to loading an input data term ~~sample~~ and filter coefficients and repeating.

6. (Currently Amended) The method of claim 3, comprising:

initializing a loop index by zero;

loading an input data term ~~sample~~ and filter coefficients;

multiplying the input data term ~~sample~~ and filter coefficient;

updating a current output;

incrementing the loop index by a number of processors being used for the computation; and

if a loop termination condition satisfied then:

stopping;

else, returning to loading an input data term ~~sample~~ and filter coefficients and repeating.

7. (Currently Amended) A finite impulse response (FIR) filter architecture, comprising:

a plurality of processors arranged in a parallel processing architecture wherein the processors share a common instruction cache and memory, the memory storing a stream of input data to be filtered and filtering coefficients for implementing the FIR filter;

an execution program that distributes computational load for FIR filtering of the input data across the plurality of processors by assigning ~~assigns~~ to each ~~available~~ processor for a given term of the input data from the stream the computation of certain specified unique partial product terms and ~~causes~~ the accumulation of ~~each~~ computed partial products for that input data to generate ~~product on specified~~ output sample values;

wherein the partial product terms comprise a product of a certain one of a plurality of FIR filter coefficients and the given term of the input data.

8. (Currently Amended) The architecture of claim 7 wherein ~~the execution program causes~~ partial product terms computed during a given computation iteration are ~~sample to be~~ reused during a later computation iteration ~~sample~~.

9. (Currently Amended) The architecture of claim 8 wherein the computed partial product terms are stored in the shared memory after use during the given computation iteration ~~give sample~~ and later retrieved therefrom by the processors for use during the later computation iteration ~~sample~~.

10. (New) A symmetric Finite Impulse Response (FIR) filter providing linear scalability, comprising:

a multiprocessor architecture wherein each processor includes an ALU (Arithmetic and Logic Unit), a Multiplier unit, Data cache, and Load/Store unit, the multiprocessor architecture sharing a common Instruction cache and a multi-port memory, and

assigning means for distributing computational load for the FIR filtering of input data across the multiprocessor architecture by assigning to each processor the computation of specified unique partial product terms and the accumulation of computed partial product terms to generate filtered output sample values;

wherein the partial product terms comprise a product of a certain one of a plurality of filter coefficients and a certain term of the input data, computed partial product terms from one computation iteration being reused by the processors in a subsequent computation iteration; and

wherein each processor in the multiprocessor architecture has access to all terms of the input data and all of the filter coefficients which are stored in the multi-port memory.

11. (New) A method for symmetric Finite Impulse Response (FIR) filtering of input data providing linear scalability using a multiprocessor architecture platform, comprising:

storing in a memory shared by plural processors in the multiprocessor architecture platform all terms of the input data to be FIR filtered and all filter coefficients;

distributing computational load for the FIR filtering of input data across the multiprocessor architecture platform by assigning to each processor the computation of specified unique partial product terms and the accumulation of computed partial product terms to generate filtered output sample values;

reusing by the processors of computed partial product terms from one computation iteration in a subsequent computation iteration;

wherein the partial product terms comprise a product of a certain one of the filter coefficients and a certain term of the input data.